

Support Vector Machines in Machine Learning

Hans D Mittelmann

Department of Mathematics and Statistics
Arizona State University

Mathematical Analysis of Large Datasets
1 May 2006

Outline

- 1 Introduction
 - What is Machine Learning?
- 2 Solving the QPs (quadratic programs)
 - The Computational Part
- 3 Three very different approaches
 - Rather concise explanations
- 4 Comparison on medium and large sets
 - REAL data! All with RBF kernel

Outline

- 1 Introduction
 - What is Machine Learning?
- 2 Solving the QPs (quadratic programs)
 - The Computational Part
- 3 Three very different approaches
 - Rather concise explanations
- 4 Comparison on medium and large sets
 - REAL data! All with RBF kernel

Which tasks in Machine Learning?

How are Support Vector Machines used?

We consider classification and testing of data in areas such as:

- computer processing of handwriting (USPS etc)
- speech recognition
- identification of faces, irises etc
- spam filtering
- categorization of newspaper articles
- analysis of medical or experimental data

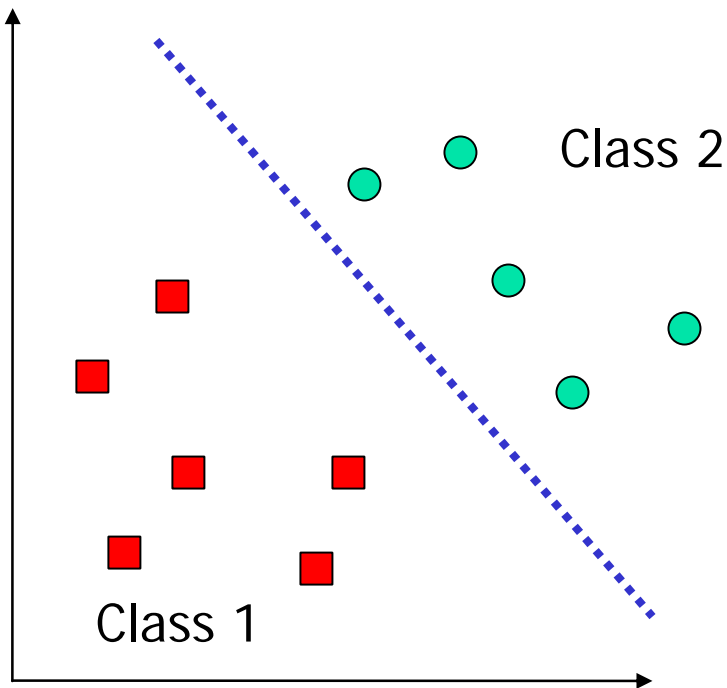
We borrowed the following introductory slides:



History of SVM

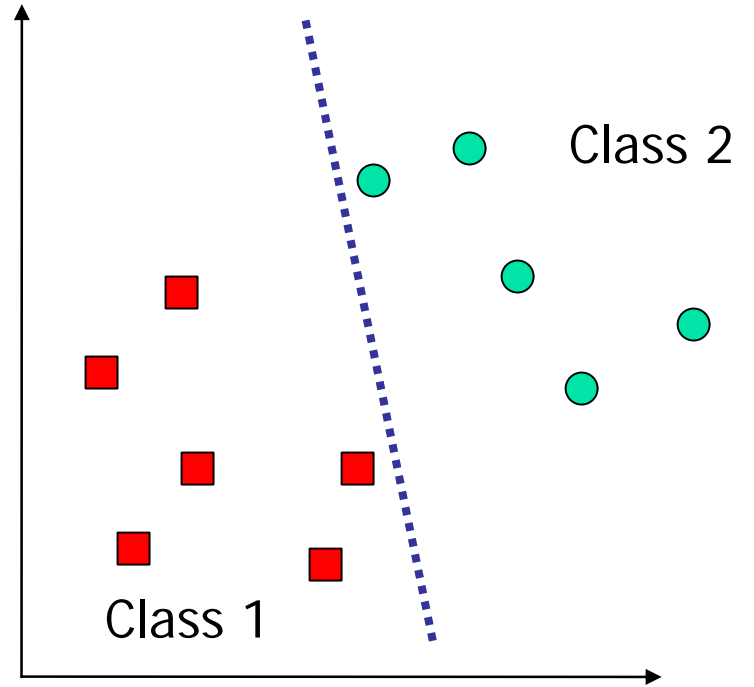
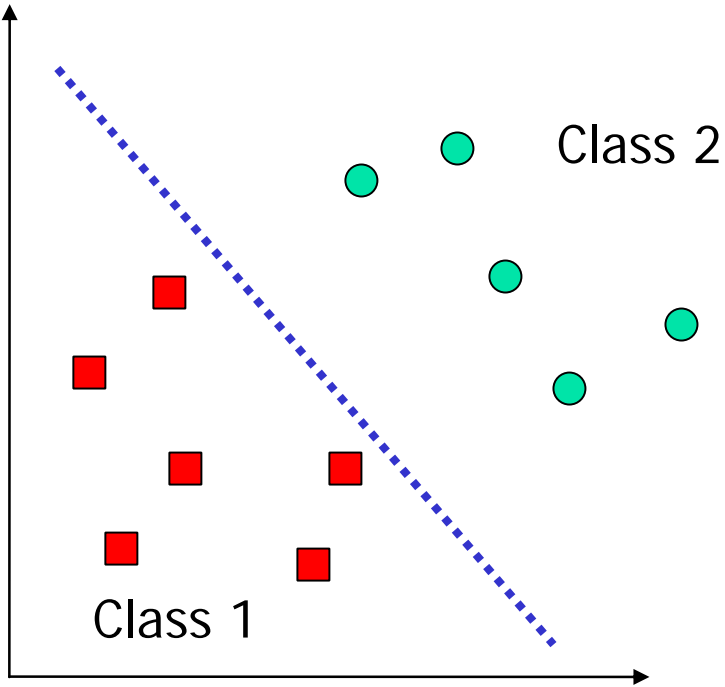
- SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis
- SVM was first introduced in COLT-92
- SVM becomes famous when, using pixel maps as input, it gives accuracy comparable to sophisticated neural networks with elaborated features in a handwriting recognition task
- Currently, SVM is closely related to:
 - Kernel methods, large margin classifiers, reproducing kernel Hilbert space, Gaussian process

Two Class Problem: Linear Separable Case



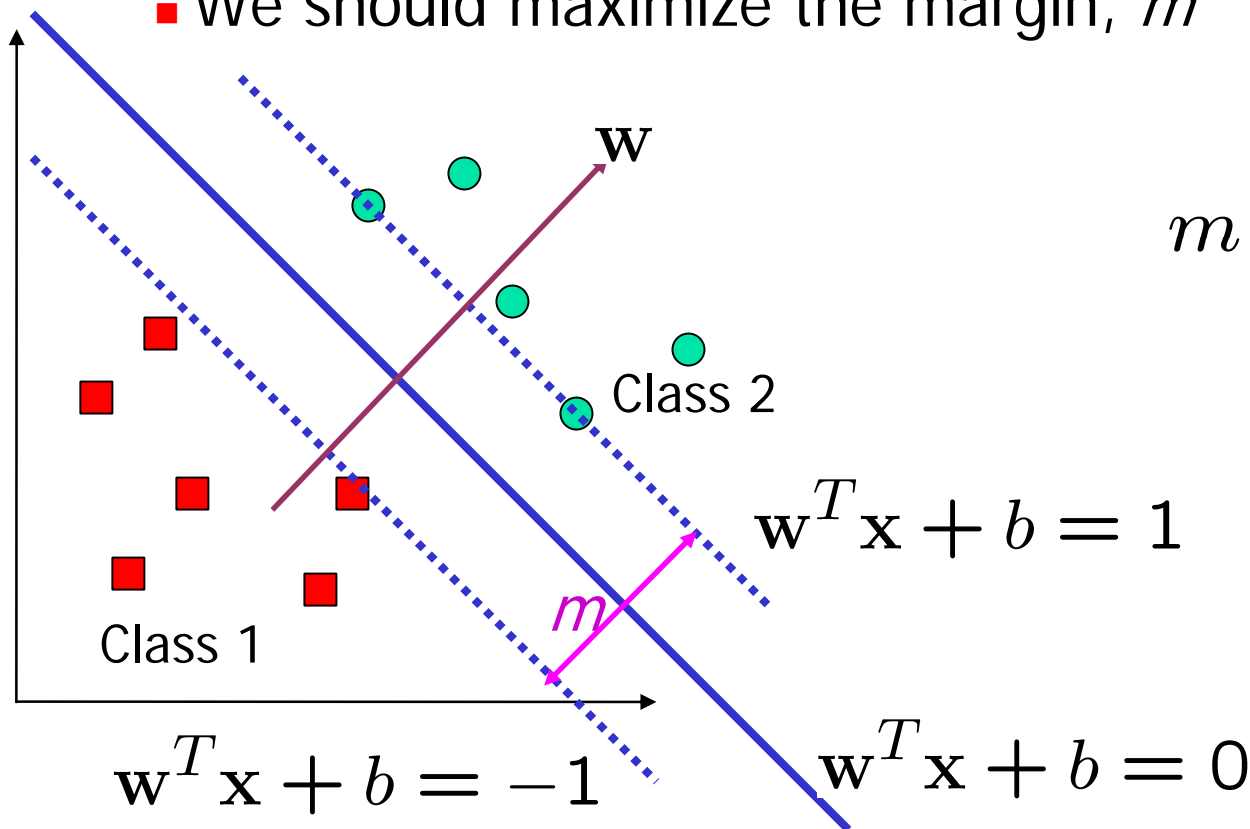
- Many decision boundaries can separate these two classes
- Which one should we choose?

Example of Bad Decision Boundaries



Good Decision Boundary: Margin Should Be Large

- The decision boundary should be as far away from the data of both classes as possible
 - We should maximize the margin, m



$$m = \frac{2}{\|w\|}$$



The Optimization Problem

- Let $\{x_1, \dots, x_n\}$ be our data set and let $y_i \in \{1, -1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \forall i$
- A constrained optimization problem

$$\text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \forall i$$

The Optimization Problem

- We can transform the problem to its dual

$$\max. W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

- This is a quadratic programming (QP) problem
 - Global maximum of α_i can always be found

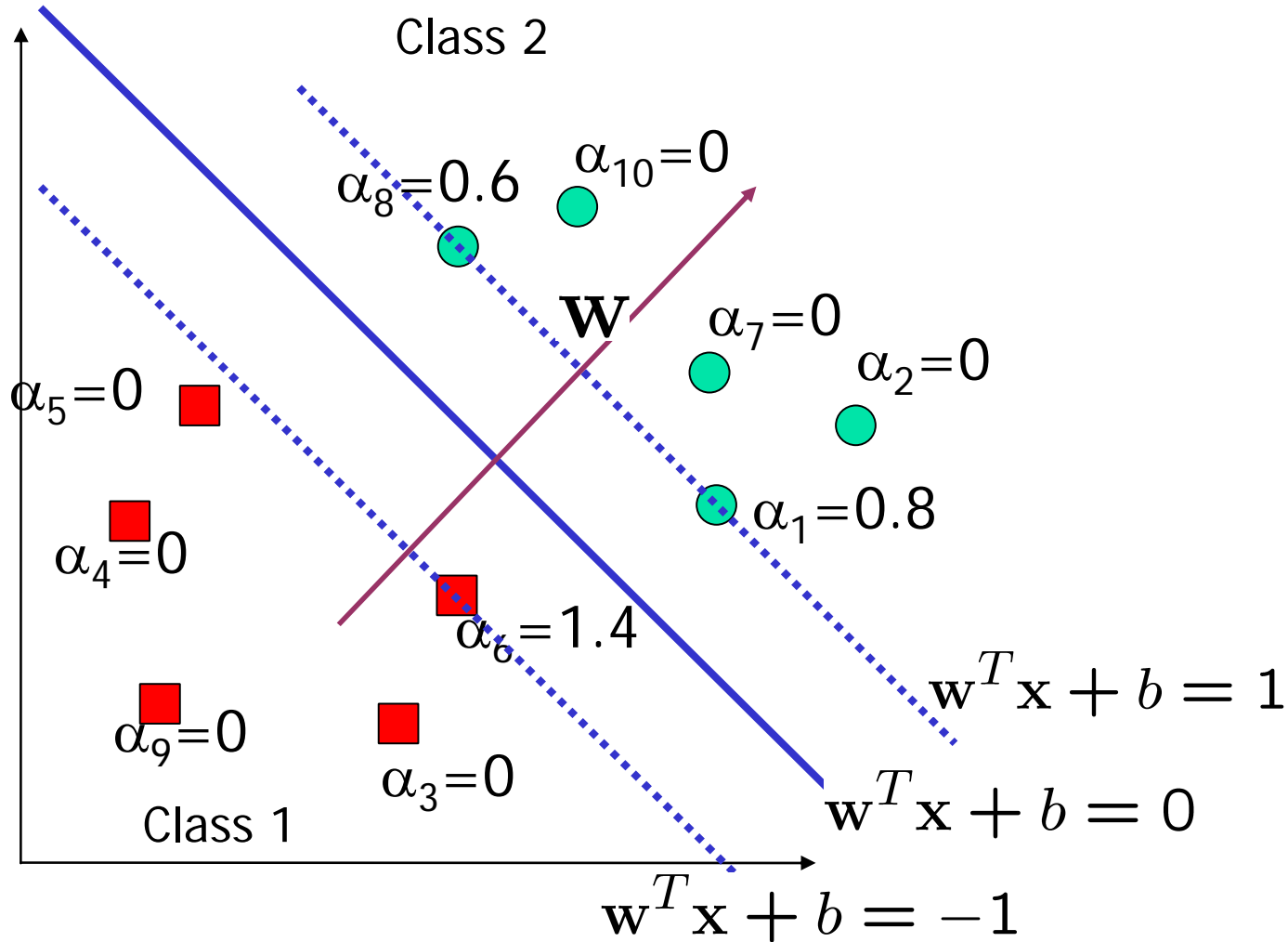
- \mathbf{w} can be recovered by $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$



Characteristics of the Solution

- Many of the α_i are zero
 - \mathbf{w} is a linear combination of a small number of data
 - Sparse representation
- \mathbf{x}_i with non-zero α_i are called support vectors (SV)
 - The decision boundary is determined only by the SV
 - Let t_j ($j=1, \dots, s$) be the indices of the s support vectors. We can write $\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- For testing with a new data \mathbf{z}
 - Compute $\mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$ and classify \mathbf{z} as class 1 if the sum is positive, and class 2 otherwise

A Geometrical Interpretation



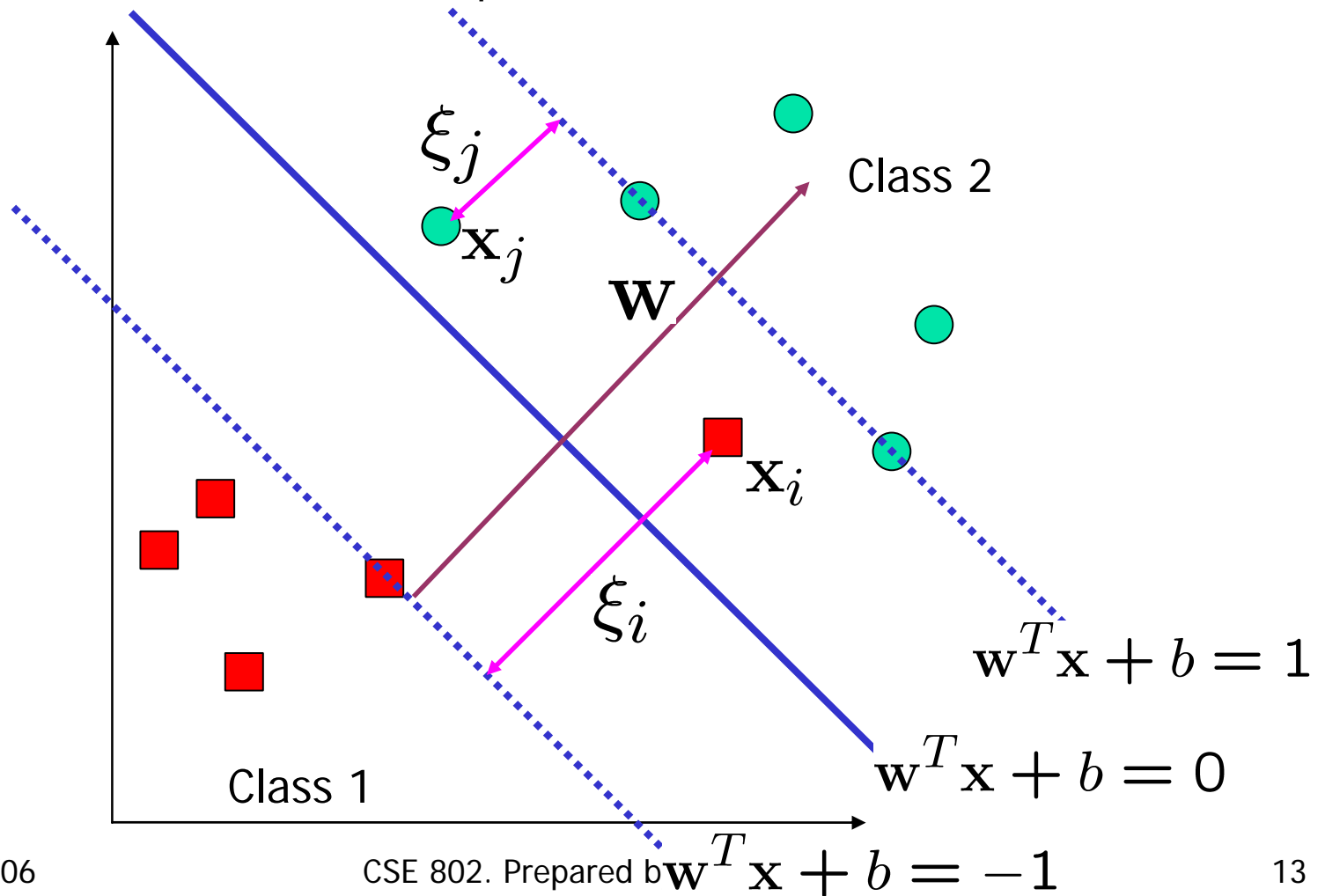


Some Notes

- There are theoretical upper bounds on the error on unseen data for SVM
 - The larger the margin, the smaller the bound
 - The smaller the number of SV, the smaller the bound
- Note that in both training and testing, the data are referenced only as inner product, $\mathbf{x}^T \mathbf{y}$
 - This is important for generalizing to the non-linear case

How About Not Linearly Separable

- We allow "error" ξ_i in classification



Soft Margin Hyperplane

- Define $\xi_i = 0$ if there is no error for x_i
 - ξ_i are just “slack variables” in optimization theory

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

- We want to minimize $\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$
 - C : tradeoff parameter between error and margin
- The optimization problem becomes

$$\text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$$

$$\text{subject to } y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

The Optimization Problem

- The dual of the problem is

$$\max. W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

- \mathbf{w} is also recovered as $\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- The only difference with the linear separable case is that there is an upper bound C on α_i
- Once again, a QP solver can be used to find α_i

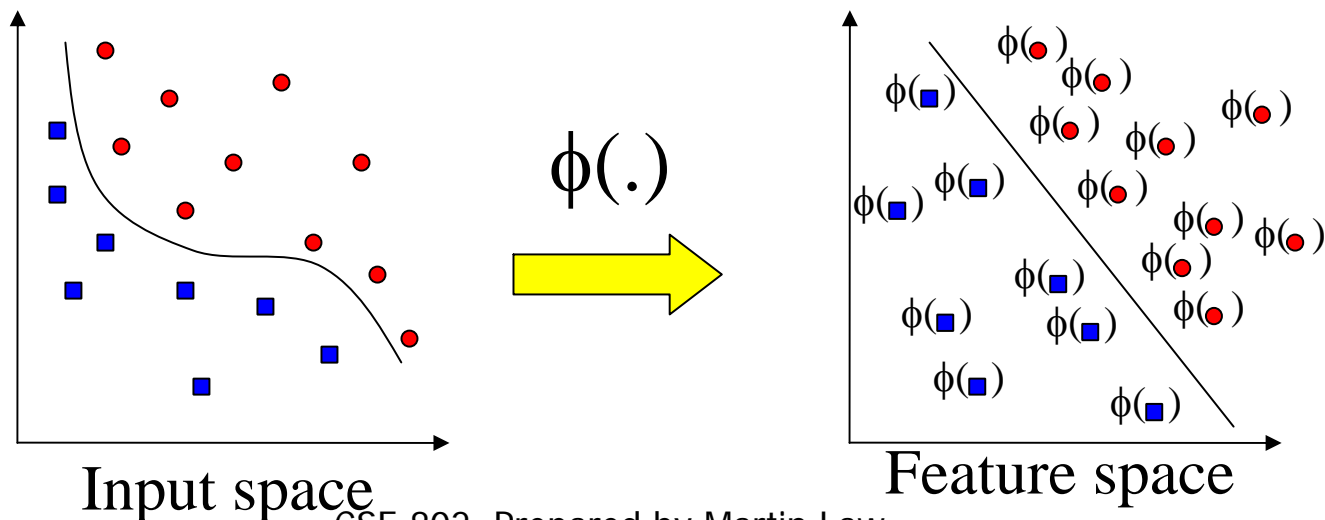


Extension to Non-linear Decision Boundary

- Key idea: transform \mathbf{x}_i to a higher dimensional space to “make life easier”
 - Input space: the space \mathbf{x}_i are in
 - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
 - Linear operation in the feature space is equivalent to non-linear operation in input space
 - The classification task can be “easier” with a proper transformation. Example: XOR

Extension to Non-linear Decision Boundary

- Possible problem of the transformation
 - High computation burden and hard to get a good estimate
- SVM solves these two issues simultaneously
 - Kernel tricks for efficient computation
 - Minimize $\|\mathbf{w}\|^2$ can lead to a “good” classifier



Example Transformation

- Define the kernel function $K(\mathbf{x}, \mathbf{y})$ as

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- Consider the following transformation

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\begin{aligned} \langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \rangle &= (1 + x_1y_1 + x_2y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

- The inner product can be computed by K without going through the map $\phi(\cdot)$



Kernel Trick

- The relationship between the kernel function K and the mapping $\phi(\cdot)$ is

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

- This is known as the kernel trick
- In practice, we specify K , thereby specifying $\phi(\cdot)$ indirectly, instead of choosing $\phi(\cdot)$
- Intuitively, $K(\mathbf{x}, \mathbf{y})$ represents our desired notion of similarity between data \mathbf{x} and \mathbf{y} and this is from our prior knowledge
- $K(\mathbf{x}, \mathbf{y})$ needs to satisfy a technical condition (Mercer condition) in order for $\phi(\cdot)$ to exist



Examples of Kernel Functions

- Polynomial kernel with degree d

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2))$$

- Closely related to radial basis function neural networks

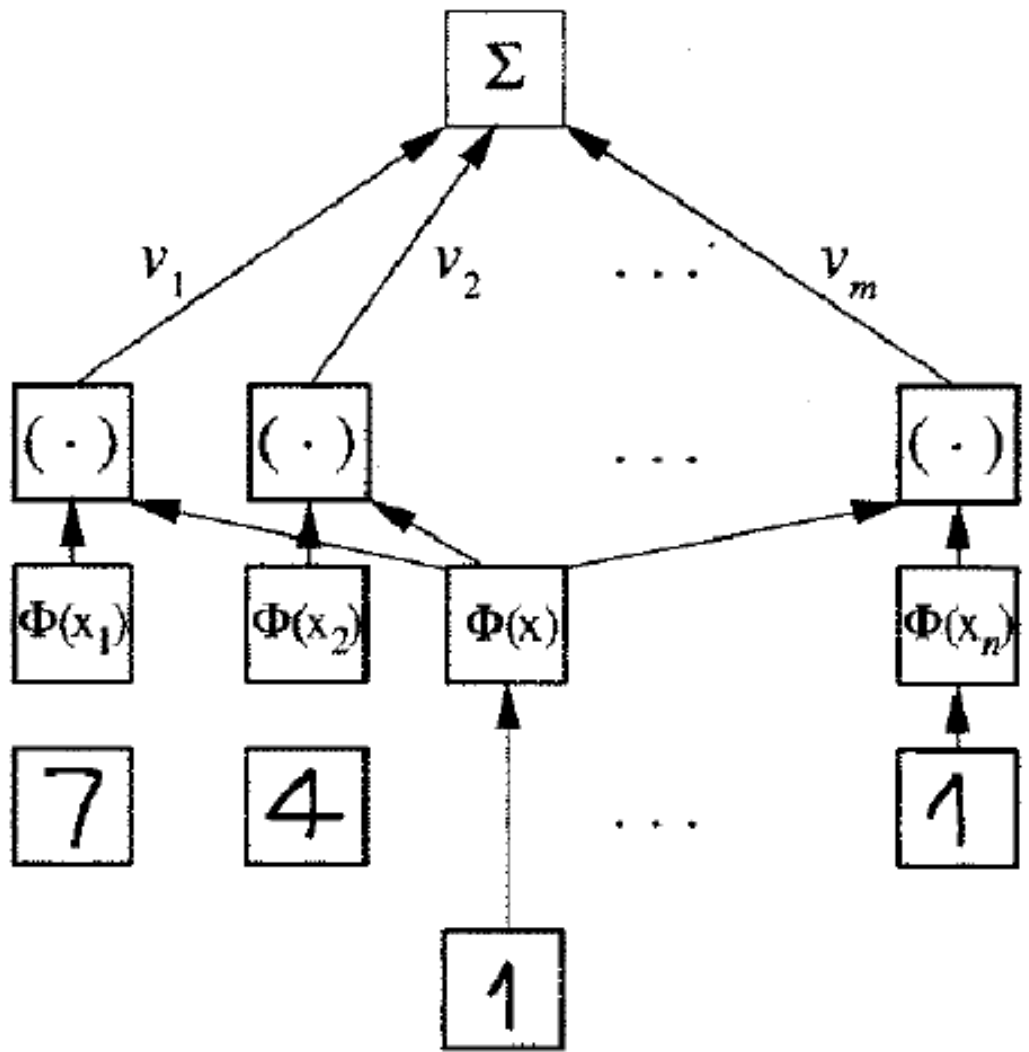
- Sigmoid with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x}^T \mathbf{y} + \theta)$$

- It does not satisfy the Mercer condition on all κ and θ

- Research on different kernel functions in different applications is very active

Example of SVM Applications: Handwriting Recognition



output $\Sigma v_i k(x, x_i) + b$

weights

dot product $(\Phi(x) \cdot \Phi(x_i)) = k(x, x_i)$

mapped vectors $\Phi(x_i), \Phi(x)$

support vectors $x_1 \dots x_n$

test vector x



Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

Original

$$\max. W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

With kernel function

$$\max. W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$



Modification Due to Kernel Function

- For testing, the new data \mathbf{z} is classified as class 1 if $f \geq 0$, and as class 2 if $f < 0$

Original

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

Example

- Suppose we have 5 1D data points
 - $x_1=1, x_2=2, x_3=4, x_4=5, x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1$
- We use the polynomial kernel of degree 2
 - $K(x,y) = (xy+1)^2$
 - C is set to 100
- We first find α_i ($i=1, \dots, 5$) by

$$\max. \sum_{i=1}^5 \alpha_i - \frac{1}{2} \sum_{i=1}^5 \sum_{j=1}^5 \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^5 \alpha_i y_i = 0$$



Example

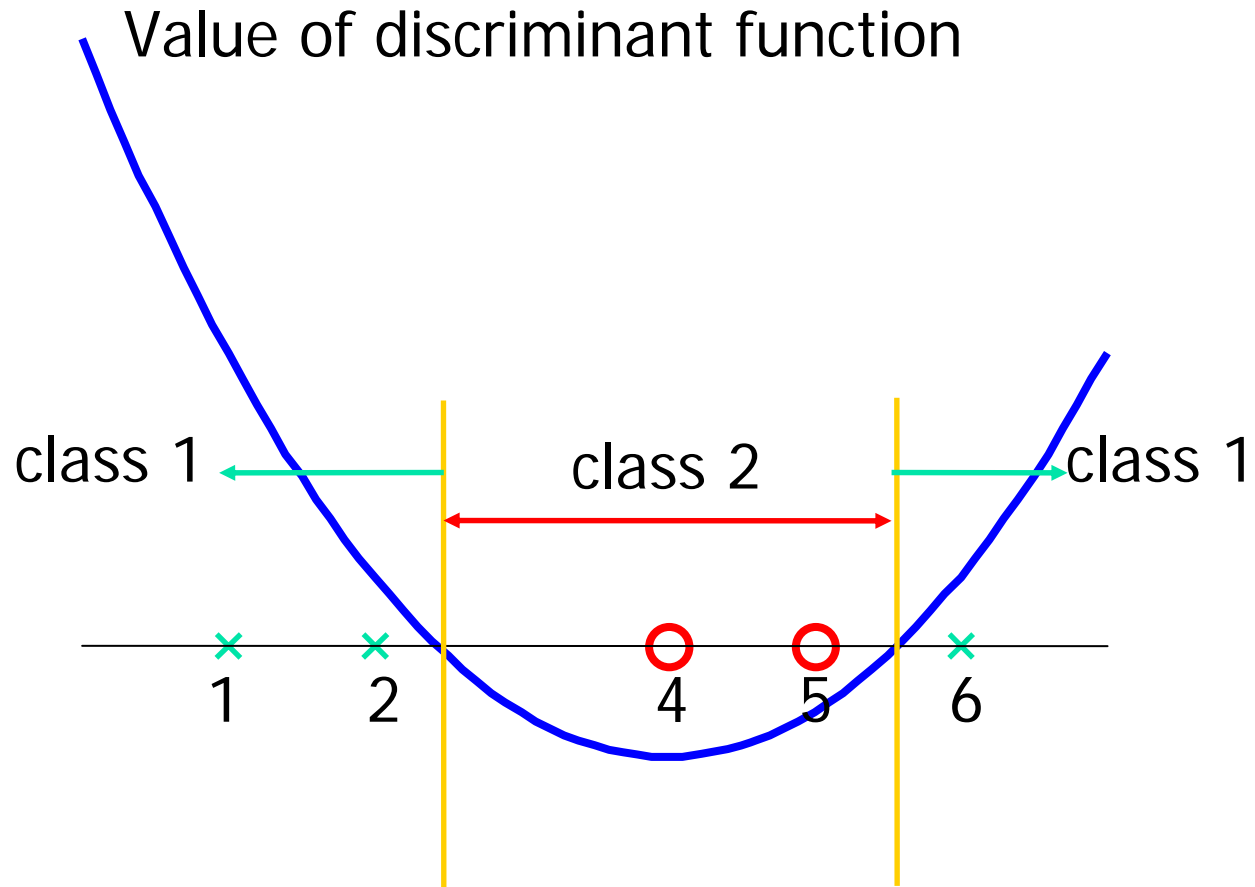
- By using a QP solver, we get
 - $\alpha_1=0, \alpha_2=2.5, \alpha_3=0, \alpha_4=7.333, \alpha_5=4.833$
 - Note that the constraints are indeed satisfied
 - The support vectors are $\{x_2=2, x_4=5, x_5=6\}$
- The discriminant function is

$$\begin{aligned} f(y) &= 2.5(1)(2y + 1)^2 + 7.333(-1)(5y + 1)^2 + 4.833(1)(6y + 1)^2 + b \\ &= 0.6667x^2 - 5.333x + b \end{aligned}$$

- b is recovered by solving $f(2)=1$ or by $f(5)=-1$ or by $f(6)=1$, as x_2, x_4, x_5 lie on $y_i(\mathbf{w}^T \phi(z) + b) = 1$ and all give $b=9$

→ $f(y) = 0.6667x^2 - 5.333x + 9$

Example





Multi-class Classification

- SVM is basically a two-class classifier
- One can change the QP formulation to allow multi-class classification
- More commonly, the data set is divided into two parts “intelligently” in different ways and a separate SVM is trained for each way of division
- Multi-class classification is done by combining the output of all the SVM classifiers
 - Majority rule
 - Error correcting code
 - Directed acyclic graph

Outline

- 1 Introduction
 - What is Machine Learning?
- 2 Solving the QPs (quadratic programs)
 - **The Computational Part**
- 3 Three very different approaches
 - Rather concise explanations
- 4 Comparison on medium and large sets
 - REAL data! All with RBF kernel

Are they standard optimization problems?

Yes, but size poses problems

Generic form of the (dual) QP

$$\min \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

subject to

$$y^T \alpha = 0$$

$$0 \leq \alpha \leq c$$

Here y binary labels, Q spd kernel matrix, c penalty for errors

This problem is **convex**

Solving the QP

It can be huge

Method of choice for solving large (but not huge) QPs with sparse matrix Q

- interior point methods

Method of choice for solving medium size QPs with dense matrix Q

- active set methods (QP extension of Simplex)

For SVM QPs are large and dense

- challenge for both methods

Solving the QP

It can be huge

Method of choice for solving large (but not huge) QPs with sparse matrix Q

- interior point methods

Method of choice for solving medium size QPs with dense matrix Q

- active set methods (QP extension of Simplex)

For SVM QPs are large and dense

- challenge for both methods

Solving the QP

It can be huge

Method of choice for solving large (but not huge) QPs with sparse matrix Q

- interior point methods

Method of choice for solving medium size QPs with dense matrix Q

- active set methods (QP extension of Simplex)

For SVM QPs are large and dense

- challenge for both methods

Solving the QP

It can be huge

Method of choice for solving large (but not huge) QPs with sparse matrix Q

- interior point methods

Method of choice for solving medium size QPs with dense matrix Q

- active set methods (QP extension of Simplex)

For SVM QPs are large and dense

- challenge for both methods

Outline

- 1 Introduction
 - What is Machine Learning?
- 2 Solving the QPs (quadratic programs)
 - The Computational Part
- 3 **Three very different approaches**
 - **Rather concise explanations**
- 4 Solving the QPs (quadratic programs)
 - REAL data! All with RBF kernel

All can deal with large cases

SVMlight (Joachims, Cornell)

- heuristic to choose small set (10) of variables to vary

SVM-QP (Scheinberg, IBM Watson RC)

- Special implementation of Simplex for SVM-QP

Core-SVM (Tsang, Kwok, Cheung, Hongkong U Sci Technol)

- Utilizing MEB (minimal enclosing ball) algorithm
- approximate but problem could be huge

All can deal with large cases

SVMLight (Joachims, Cornell)

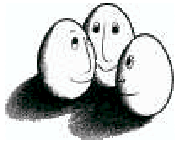
- heuristic to choose small set (10) of variables to vary

SVM-QP (Scheinberg, IBM Watson RC)

- Special implementation of Simplex for SVM-QP

Core-SVM (Tsang, Kwok, Cheung, Hongkong U Sci Technol)

- Utilizing MEB (minimal enclosing ball) algorithm
- approximate but problem could be huge

SVM^{light}

Support Vector Machine

Author: [Thorsten Joachims](mailto:thorsten@joachims.org) <thorsten@joachims.org>
[Cornell University](#)
[Department of Computer Science](#)


Developed at:
[University of Dortmund](#), [Informatik](#), [AI-Unit](#)
[Collaborative Research Center on 'Complexity Reduction in Multivariate Data'](#) (SFB475)

Version: 6.01
 Date: 02.09.2004

Overview

SVM^{light} is an implementation of Support Vector Machines (SVMs) in C. The main features of the program are the following:

- fast optimization algorithm
 - working set selection based on steepest feasible descent
 - "shrinking" heuristic
 - caching of kernel evaluations
 - use of folding in the linear case
- solves classification and regression problems. For multivariate and structured outputs use [SVM^{struct}](#).
- solves ranking problems (e. g. learning retrieval functions in [STRIVER](#) search engine).
- computes XiAlpha-estimates of the error rate, the precision, and the recall
- efficiently computes Leave-One-Out estimates of the error rate, the precision, and the recall
- includes algorithm for approximately training large transductive SVMs (TSVMs) (see also [Spectral Graph Transducer](#))
- can train SVMs with cost models and example dependent costs
- allows restarts from specified vector of dual variables
- handles many thousands of support vectors
- handles several hundred-thousands of training examples
- supports standard kernel functions and lets you define your own
- uses sparse vector representation

 [SVM^{struct}](#): SVM learning for multivariate and structured outputs like trees, sequences, and sets (available [here](#)).

Description

SVM^{light} is an implementation of Vapnik's Support Vector Machine [[Vapnik, 1995](#)] for the problem of pattern recognition, for the problem of regression, and for the problem of learning a ranking function. The optimization algorithms used in SVM^{light} are described in [[Joachims, 2002a](#)]. [[Joachims, 1999a](#)]. The algorithm has scalable memory requirements and can handle problems with many thousands of support vectors efficiently.

The software also provides methods for assessing the generalization performance efficiently. It includes two efficient estimation methods for both error rate and precision/recall. XiAlpha-estimates [[Joachims, 2002a](#), [Joachims, 2000b](#)] can be computed at essentially no computational expense, but they are conservatively biased. Almost unbiased estimates provides leave-one-out testing. SVM^{light} exploits that the results of most leave-one-outs (often more than 99%) are predetermined and need not be computed [[Joachims, 2002a](#)].

New in this version is an algorithm for learning ranking functions [[Joachims, 2002c](#)]. The goal is to learn a function from preference examples, so that it orders a new set of objects as accurately as possible. Such ranking problems naturally occur in applications like search engines and recommender systems.

Futhermore, this version includes an algorithm for training large-scale transductive SVMs. The algorithm proceeds by solving a sequence of optimization problems lower-bounding the solution using a form of local search. A detailed description of the algorithm can be found in [[Joachims, 1999c](#)]. A similar transductive learner, which can be thought of as a

All can deal with large cases

SVMLight (Joachims, Cornell)

- heuristic to choose small set (10) of variables to vary

SVM-QP (Scheinberg, IBM Watson RC)

- Special implementation of Simplex for SVM-QP

Core-SVM (Tsang, Kwok, Cheung, Hongkong U Sci Technol)

- Utilizing MEB (minimal enclosing ball) algorithm
- approximate but problem could be huge

An Efficient Implementation of an Active Set Method for SVM

Katya Scheinberg
IBM T. J. Watson Research Center
katyas@us.ibm.com

September 30, 2005

Abstract

We propose an active set algorithm to solve the convex quadratic programming (QP) problem which is the core of the support vector machine (SVM) training. The underlying method is not new and is based on the extensive practice of the Simplex method and its variants for convex quadratic problems. However, its application to large-scale SVM problems is new. Until recently the traditional active set methods were considered impractical for large SVM problems. By adapting the methods to the special structure of SVM problems we were able to produce an efficient implementation. We conduct an extensive study of the behavior of our method and its variations on SVM problems. We present computational results comparing our method with Joachims' SVM^{light} [16]. The results show that our method has overall better performance on many SVM problems. It seems to have particularly strong advantage on more difficult problems. In addition this algorithm has better theoretical properties and it naturally extends to the incremental mode.

1 Introduction

In this paper we introduce an active set method to solve the following convex quadratic programming (QP) optimization problem which is defined by *1-Norm Soft Margin SVM* problem.

$$(P) \quad \begin{array}{ll} \max & -\frac{1}{2}\alpha^T Q\alpha - c^T \xi \\ \text{s.t.} & -Q\alpha + by + s - \xi = -e, \end{array}$$

All can deal with large cases

SVMLight (Joachims, Cornell)

- heuristic to choose small set (10) of variables to vary

SVM-QP (Scheinberg, IBM Watson RC)

- Special implementation of Simplex for SVM-QP

Core-SVM (Tsang, Kwok, Cheung, Hongkong U Sci Technol)

- Utilizing MEB (minimal enclosing ball) algorithm
- approximate but problem could be huge

Core Vector Machines: Fast SVM Training on Very Large Data Sets

Ivor W. Tsang
James T. Kwok
Pak-Ming Cheung

*Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay
Hong Kong*

IVOR@CS.UST.HK
JAMESK@CS.UST.HK
PAKMING@CS.UST.HK

Editor: Nello Cristianini

Abstract

Standard SVM training has $O(m^3)$ time and $O(m^2)$ space complexities, where m is the training set size. It is thus computationally infeasible on very large data sets. By observing that practical SVM implementations only *approximate* the optimal solution by an iterative strategy, we scale up kernel methods by exploiting such “approximateness” in this paper. We first show that many kernel methods can be equivalently formulated as minimum enclosing ball (MEB) problems in computational geometry. Then, by adopting an efficient approximate MEB algorithm, we obtain provably approximately optimal solutions with the idea of core sets. Our proposed Core Vector Machine (CVM) algorithm can be used with nonlinear kernels and has a time complexity that is *linear* in m and a space complexity that is *independent* of m . Experiments on large toy and real-world data sets demonstrate that the CVM is as accurate as existing SVM implementations, but is much faster and can handle much larger data sets than existing scale-up methods. For example, CVM with the Gaussian kernel produces superior results on the KDDCUP-99 intrusion detection data, which has about five million training patterns, in only 1.4 seconds on a 3.2GHz Pentium-4 PC.

Keywords: kernel methods, approximation algorithm, minimum enclosing ball, core set, scalability

1. Introduction

In recent years, there has been a lot of interest on using kernels in various machine learning problems, with the support vector machines (SVM) being the most prominent example. Many of these kernel methods are formulated as quadratic programming (QP) problems. Denote the number of training patterns by m . The training time complexity of QP is $O(m^3)$ and its space complexity is at least quadratic. Hence, a major stumbling block is in scaling up these QP’s to large data sets, such as those commonly encountered in data mining applications.

To reduce the time and space complexities, a popular technique is to obtain low-rank approximations on the kernel matrix, by using the Nyström method (Williams and Seeger, 2001), greedy approximation (Smola and Schölkopf, 2000), sampling (Achlioptas et al., 2002) or matrix decompositions (Fine and Scheinberg, 2001). However, on very large data sets, the resulting rank of the kernel matrix may still be too high to be handled efficiently.

Outline

- 1 Introduction
 - What is Machine Learning?
- 2 Solving the QPs (quadratic programs)
 - The Computational Part
- 3 Three very different approaches
 - Rather concise explanations
- 4 Comparison on medium and large sets
 - REAL data! All with RBF kernel

Frequently used datasets

small, medium, large

Adult dataset (2-6 MB, depending on format)

- 32560 elements, 123 attributes
predict income >50K/year from census data

Web dataset (8-9 MB)

- 49749 elements, 300 attributes
log of anonymous visitors of www.microsoft.com

USPS dataset (500-600 MB)

- 266079 elements, 675 attributes
handwriting data from USPS

Just show results for learning. Testing was done also,

Frequently used datasets

small, medium, large

Adult dataset (2-6 MB, depending on format)

- 32560 elements, 123 attributes
predict income >50K/year from census data

Web dataset (8-9 MB)

- 49749 elements, 300 attributes
log of anonymous visitors of www.microsoft.com

USPS dataset (500-600 MB)

- 266079 elements, 675 attributes
handwriting data from USPS

Just show results for learning. Testing was done also,

Frequently used datasets

small, medium, large

Adult dataset (2-6 MB, depending on format)

- 32560 elements, 123 attributes
predict income >50K/year from census data

Web dataset (8-9 MB)

- 49749 elements, 300 attributes
log of anonymous visitors of www.microsoft.com

USPS dataset (500-600 MB)

- 266079 elements, 675 attributes
handwriting data from USPS

Just show results for learning. Testing was done also,

Frequently used datasets

small, medium, large

Adult dataset (2-6 MB, depending on format)

- 32560 elements, 123 attributes
predict income >50K/year from census data

Web dataset (8-9 MB)

- 49749 elements, 300 attributes
log of anonymous visitors of www.microsoft.com

USPS dataset (500-600 MB)

- 266079 elements, 675 attributes
handwriting data from USPS

Just show results for learning. Testing was done also,

Frequently used datasets

small, medium, large

Adult dataset (2-6 MB, depending on format)

- 32560 elements, 123 attributes
predict income >50K/year from census data

Web dataset (8-9 MB)

- 49749 elements, 300 attributes
log of anonymous visitors of www.microsoft.com

USPS dataset (500-600 MB)

- 266079 elements, 675 attributes
handwriting data from USPS

Just show results for learning. Testing was done also,

Results, adult set (AMD-64, 2.4GHz)

code	params	time	SV	BSV
SVMlight	g= .1	14466	9959	3200
	g= .01	7200	1703	9783
	g= .001	937	196	11361
SVM-QP	sh=10			
	sh=100	460	1317	9953
	sh=1000	278	143	11384
CVM	g=1e-1	1309	9224	3353
	g=1e-2	828	1278	9879
	g=1e-3	443	190	11367

Results, web set (AMD-64, 2.4GHz)

code	params	time	SV	BSV
SVMlight	g= .1	1354	4025	495
	g= .01	3581	2097	825
	g= .001	694	437	1645
SVM-QP	sh=10	715	3446	527
	sh=100	174	1404	905
	sh=1000	92	297	1702
CVM	g=1e-1	407	3650	508
	g=1e-2	358	1458	839
	g=1e-3	266	397	1675

Results, USPS set (AMD-64, 2.4GHz)

code	params	time	SV	BSV
SVMlight	g= .01	1713	2906	0
	g= .001	1349	1371	1
	g= .0001	4308	560	3296
SVM-QP	sh=100	1591	2906	0
	sh=1000	837	1370	1
	sh=10000	5265	564	3293
CVM	g=1e-2	2145	2898	0
	g=1e-3	1142	1372	1
	g=1e-4	2118	593	3279

Observations, Future Work

We notice

- SVM-QP treats explicitly variables (active) on upper or lower bounds
- SVMlight varies very few variables at a time and convergence of variables to bounds is slow
- SVM-QP is better if many variables at bounds
- CVM is slower than SVM-QP if many SV at bounds (BSV)

Future work

- Collaborate with K. Scheinberg on development of SVM-QP

Observations, Future Work

We notice

- SVM-QP treats explicitly variables (active) on upper or lower bounds
- SVMlight varies very few variables at a time and convergence of variables to bounds is slow
- SVM-QP is better if many variables at bounds
- CVM is slower than SVM-QP if many SV at bounds (BSV)

Future work

- Collaborate with K. Scheinberg on development of SVM-QP

Observations, Future Work

We notice

- SVM-QP treats explicitly variables (active) on upper or lower bounds
- SVMlight varies very few variables at a time and convergence of variables to bounds is slow
- SVM-QP is better if many variables at bounds
- CVM is slower than SVM-QP if many SV at bounds (BSV)

Future work

- Collaborate with K. Scheinberg on development of SVM-QP

Observations, Future Work

We notice

- SVM-QP treats explicitly variables (active) on upper or lower bounds
- SVMlight varies very few variables at a time and convergence of variables to bounds is slow
- SVM-QP is better if many variables at bounds
- CVM is slower than SVM-QP if many SV at bounds (BSV)

Future work

- Collaborate with K. Scheinberg on development of SVM-QP

Observations, Future Work

We notice

- SVM-QP treats explicitly variables (active) on upper or lower bounds
- SVMlight varies very few variables at a time and convergence of variables to bounds is slow
- SVM-QP is better if many variables at bounds
- CVM is slower than SVM-QP if many SV at bounds (BSV)

Future work

- Collaborate with K. Scheinberg on development of SVM-QP

Observations, Future Work

We notice

- SVM-QP treats explicitly variables (active) on upper or lower bounds
- SVMlight varies very few variables at a time and convergence of variables to bounds is slow
- SVM-QP is better if many variables at bounds
- CVM is slower than SVM-QP if many SV at bounds (BSV)

Future work

- Collaborate with K. Scheinberg on development of SVM-QP